Modelling of Software intensive systems

Assignment 1 Gantry System

First Edition

L. Jason	Jason.Liu@student.uantwerpen.be	s0213082
S. Kin Ning	KinNing.Shum@student.uantwerpen.be	s0202054

2024 - 12 - 21

Contents

1 | Plant Model Creation

Model used for the simulations:

```
model CraneModel
  type DampingFactor = Real(unit = "N/(m/s)");
  // Parameters
  parameter Modelica.Units.SI.Mass m = 0.2 "The mass of the pendulum bob/container";
  parameter Modelica.Units.SI.Mass M = 10 "The mass of the trolley/cart";
  parameter Modelica.Units.SI.Length r = 1 "The length of the rope connecting the
trolley/cart and the pendulum bob/container";
  parameter DampingFactor d_p = 0.12 "The damping factor of the pendulum";
  parameter DampingFactor d_c = 4.7895 "The damping factor for the motion of the
trolley/cart";
  constant Modelica.Units.SI.Acceleration g = Modelica.Constants.g_n "The
acceleration due to gravity";
  // Variables
  Real u = 0 "The control signal to move the pendulum and trolley/cart";
  Modelica.Units.SI.Length x "The displacement of the trolley/cart";
  Modelica.Units.SI.Velocity v "The velocity of the trolley/cart";
  Modelica.Units.SI.Angle theta "The angular displacement of the pendulum";
  Modelica.Units.SI.AngularVelocity omega "The angular velocity of the pendulum";
initial equation
  x = 0;
  v = 0;
  theta = 0;
  omega = 0;
equation
// First equation
  der(x) = v;
// Second equation
  der(theta) = omega;
// Third equation
  der(v) = (r*(d_c*v - m*(g*sin(theta)*cos(theta) + r*omega^2*sin(theta)) - u) -
(d_p*cos(theta)*omega))/(-r*(M + m*(sin(theta))^2));
// Fourth equation
  der(omega) = ((d_p*omega*(m + M)) + (m^2*r^2*sin(theta)*cos(theta)*omega^2) +
m*r*(((g*sin(theta))*(m + M)) + (cos(theta)*(u - d_c*v))))/((m*r^2)*(-M -
(m*(sin(theta))^2)));
// u signal change
if time < 0.5 then
  u = 1000;
else
  u = 0;
end if;
end CraneModel;
```

There is a built-in type for the translational damping coefficient with unit $\frac{N \cdot s}{m}$. However, it is not entirely accurate for our model, so we made a new type instead.

1.1 | Motion of the system

Let's discuss the behavior of the default plant model, or in our case the model of the crane over time. In the following simulations, u is set to 1000 in the first 0.5 seconds, and 0 otherwise.

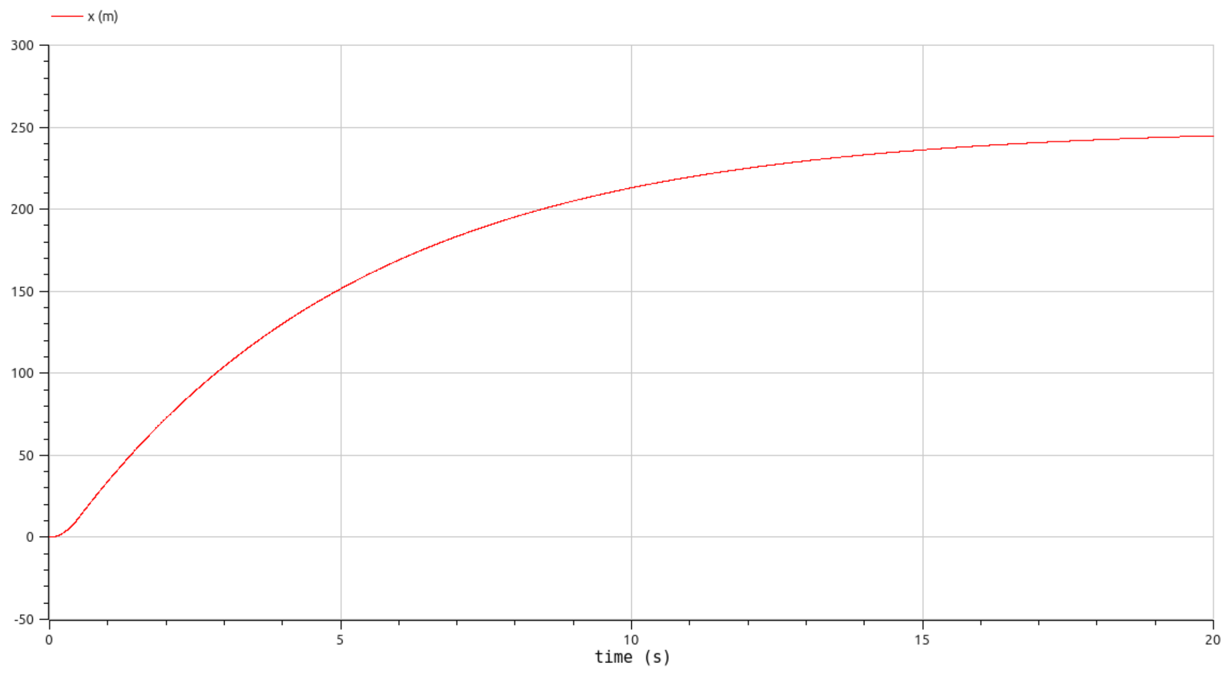

Figure 1: Displacement of the trolley[1], part_1/1_x.png

We can see that the trolley first moves quickly and slowly settles down over time.

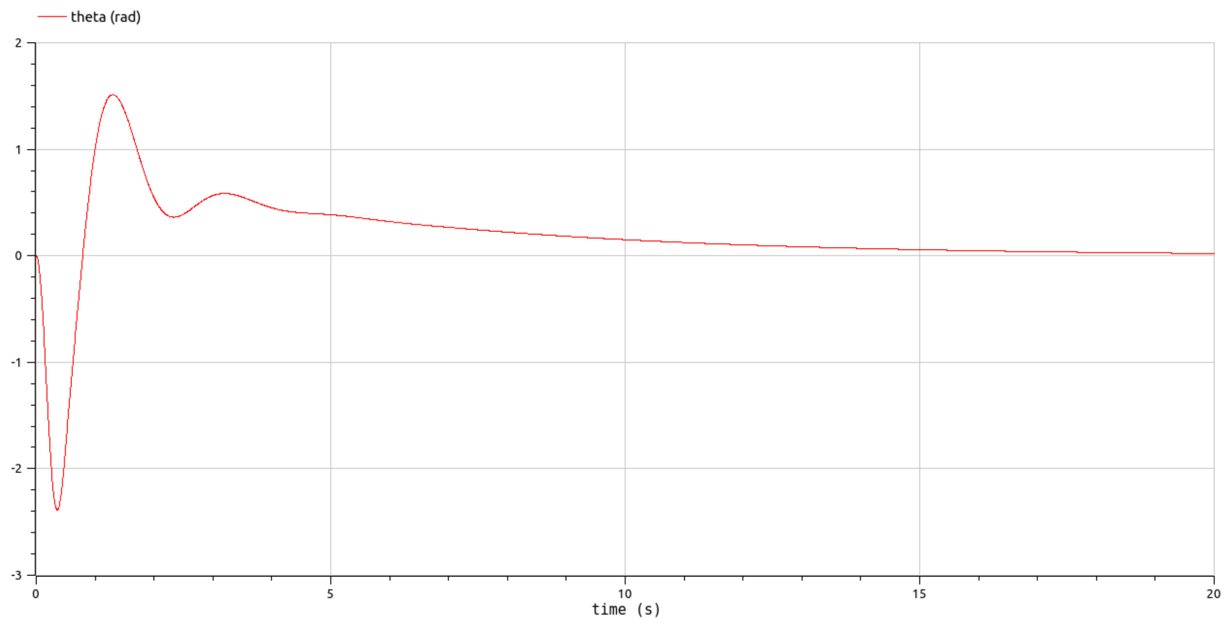

Figure 2: Angle of the pendulum, part_1/1_theta.png

We can see that the pendulum overshoots pretty hard and slowly reaches a stable angle over time.

[1]All simulations in this document are run with interval 0.004s

So here we can clearly see the effects that the control signal has. When the control signal is high, it tries hard to move to the needed position. When it becomes zero it indicates that we want to stop, so it slowly settles down till it is stable again.

2 | Plant Model Calibration

We need to find damping coefficients d_p and d_c given real life data. To do this, we have looped over possible values[2] of d_p and d_c and compared them to the calibration data. The values with the least error is the best fit. Error is calculated by squaring the sum of the difference.

2.1 | Experiment 1

Here we will look at how the d_c value change, and we will try to get the closest value to the calibration data. d_c is the damping coefficient of the cart.

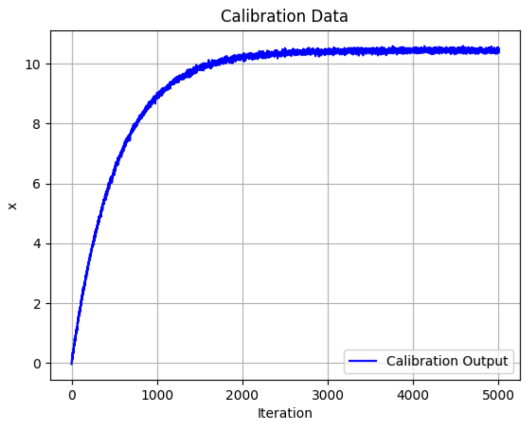

Figure 3: Simulated data with $d_c = 4.7895$, part_2/experiment_one_traces_sim.png

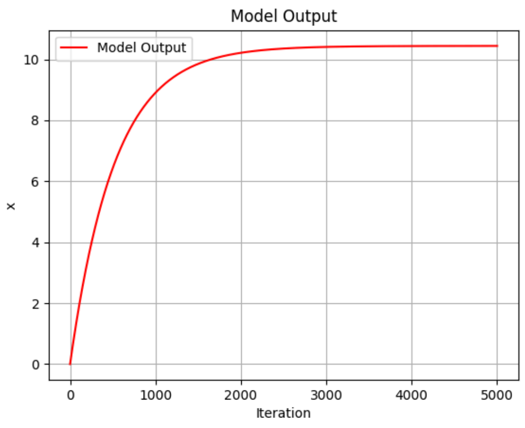

Figure 4: Real life data, part_2/experiment_one_traces_rl.png

Looking at two graphs, the data resembles each other a lot and has a square error of 12.6998. The value of d_c is 4.7895. A better comparison can be seen in the following figure.

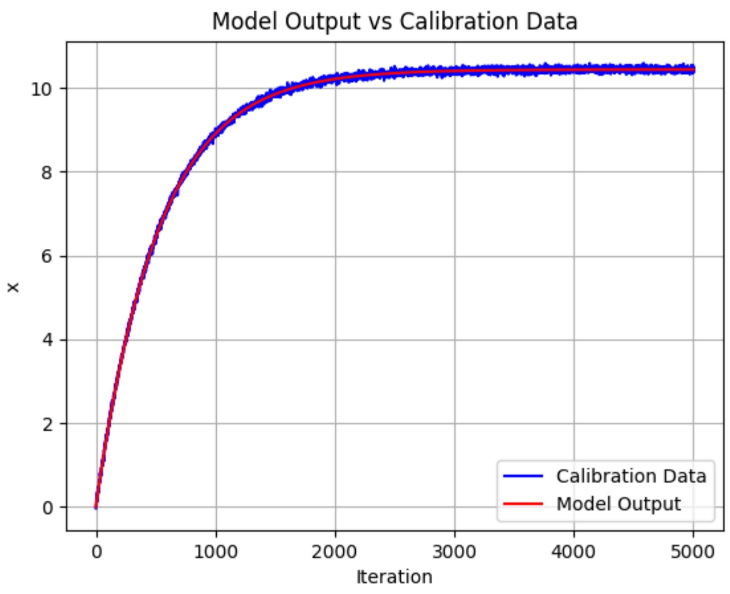

Figure 5: Combined graph, part_2/experiment_one_traces_comb.png

[2]with a step size of 0.0005

2.2 | Experiment 2

Here we will look at d_p and how different values have an impact. We will try to get the closest value to the calibration data again. d_p is the damping coefficient of the pendulum.

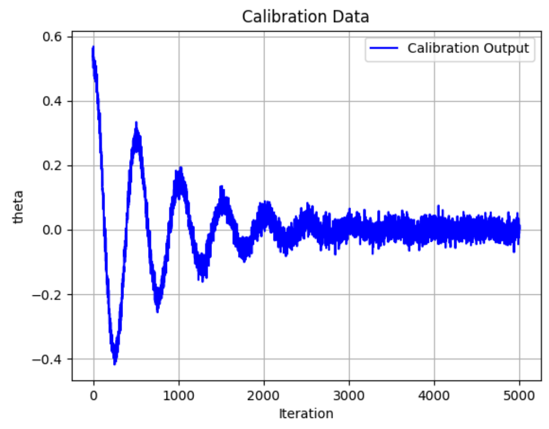

Figure 6: Simulated data with $d_c = 0.12$, part_2/experiment_two_traces_sim.png

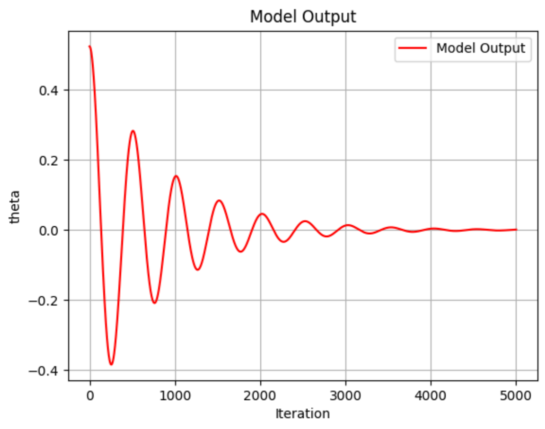

Figure 7: Real life data, part_2/experiment_two_traces_rl.png

We can see that the value with the lowest error is 0.12 for d_p, the error being 1.9574.

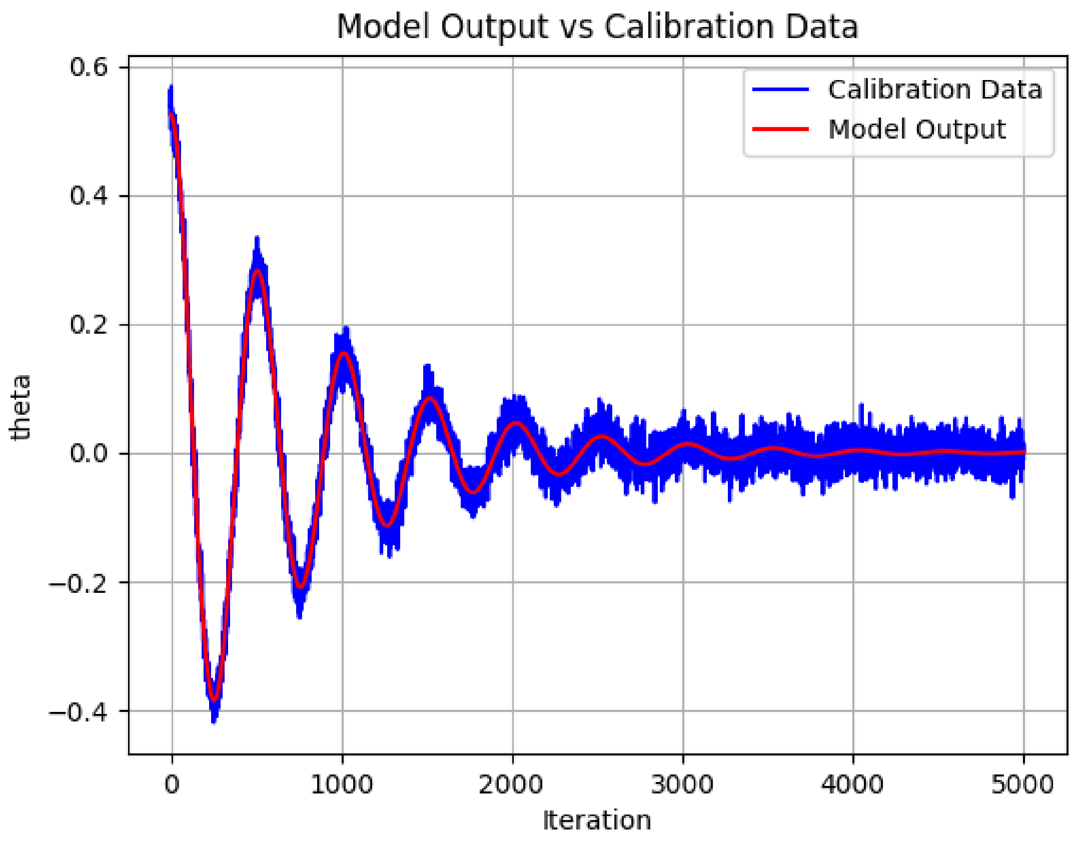

Figure 8: Combined graph, part_2/experiment_two_traces_comb.png

3 | Controller Model Creation

We have created a PID control loop and in this section, we will analyze each parameter of the control unit, namely K_p, K_i and K_d.

3.1 | Parameter K_p

We analyze K_p by setting it to 1, and the others to 0. The set point in this simulation, and for the upcoming simulations in this section, is 20 meters. This results in the following graphs:

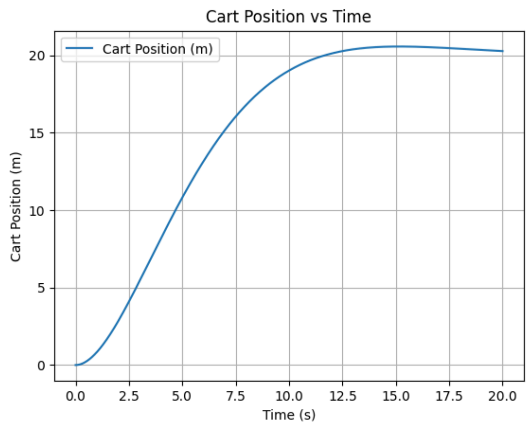

Figure 9: Cart position with K_p = 1, part_3/p1_cart_position.png

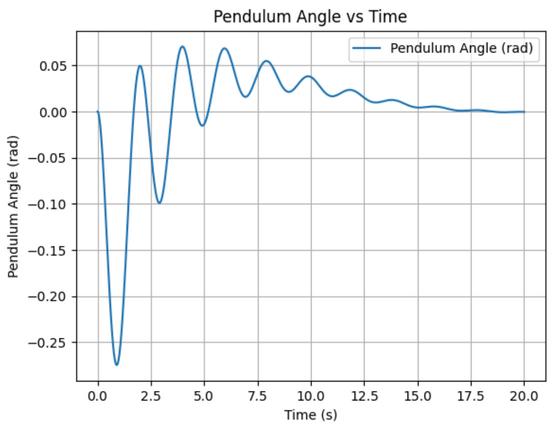

Figure 10: Pendulum angle with K_p = 1, part_3/p1_pendulum_angle.png

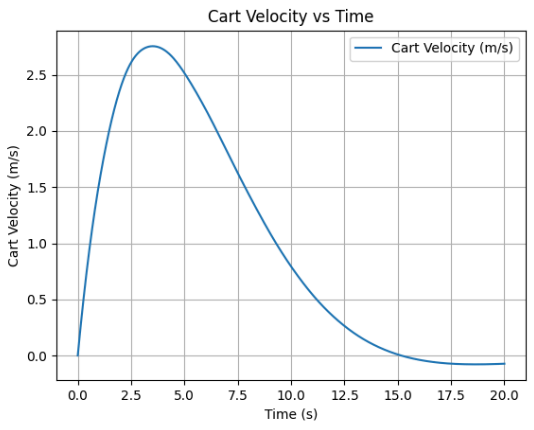

Figure 11: Cart velocity with K_p = 1, part_3/p1_cart_velocity.png

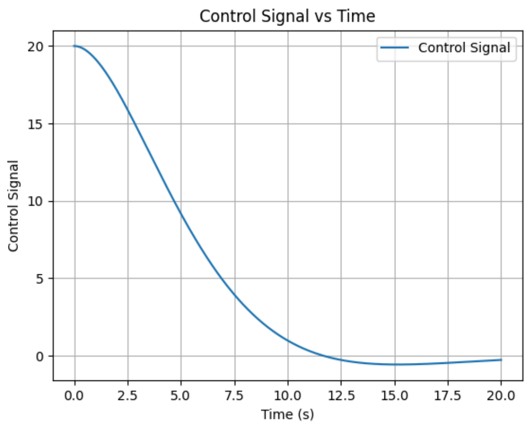

Figure 12: Control signal with K_p = 1, part_3/p1_control_signal.png

We notice that the cart velocity is highest when the cart is furthest away from the destination, and slowly decays the closer it gets to it. We also notice that the cart overshoots the destination, and has to backtrack, but since it is so close, the velocity is not very high. The pendulum angle does not bring any new information.

This makes sense, as the control signal is $u(t) = K_p(20 - x)$, so the closer we get, the velocity will also decrease. But one problem arises, and that is that when the cart reaches its destination, u will be equal to 0. What this means, is that the cart will not stop at the set point, as it keeps its momentum. This results in the cart always overshooting the destination. This effect is amplified the higher K_p is, which we will show now.

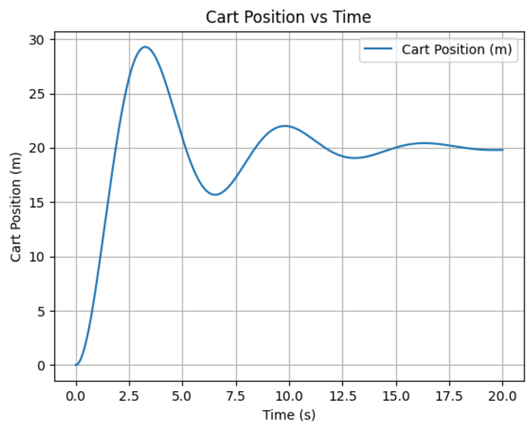

Figure 13: Cart position with K_p = 10, part_3/p10_cart_position.png

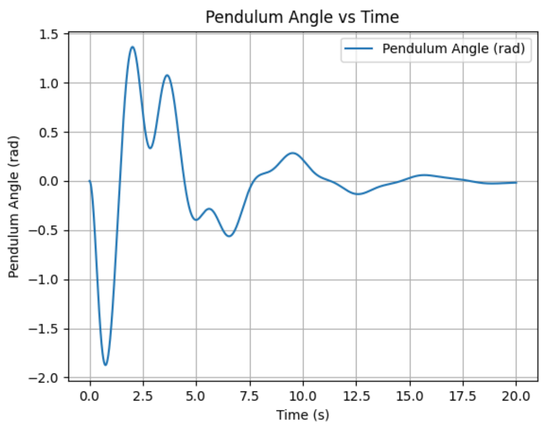

Figure 14: Pendulum angle with K_p = 10, part_3/p10_pendulum_angle.png

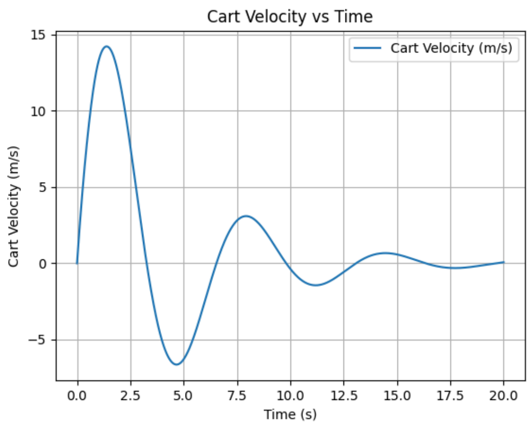

Figure 15: Cart velocity with K_p = 10, part_3/p10_cart_velocity.png

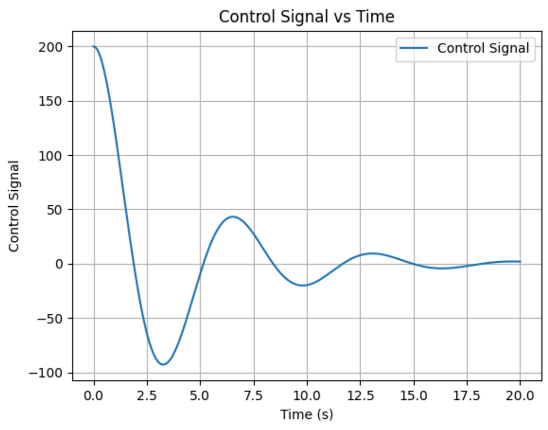

Figure 16: Control signal with K_p = 10, part_3/p10_control_signal.png

With $K_p = 10$, we see that the initial velocity is much higher, so it will reach the destination much faster. Now it reaches it in 2 seconds instead of 12 seconds in the previous simulation. However, it overshoots very hard, and when it backtracks, it also overshoots that. We can conclude that K_p alone is not a very good way to control the cart, as it is either too slow, or it overshoots and has to backtrack.

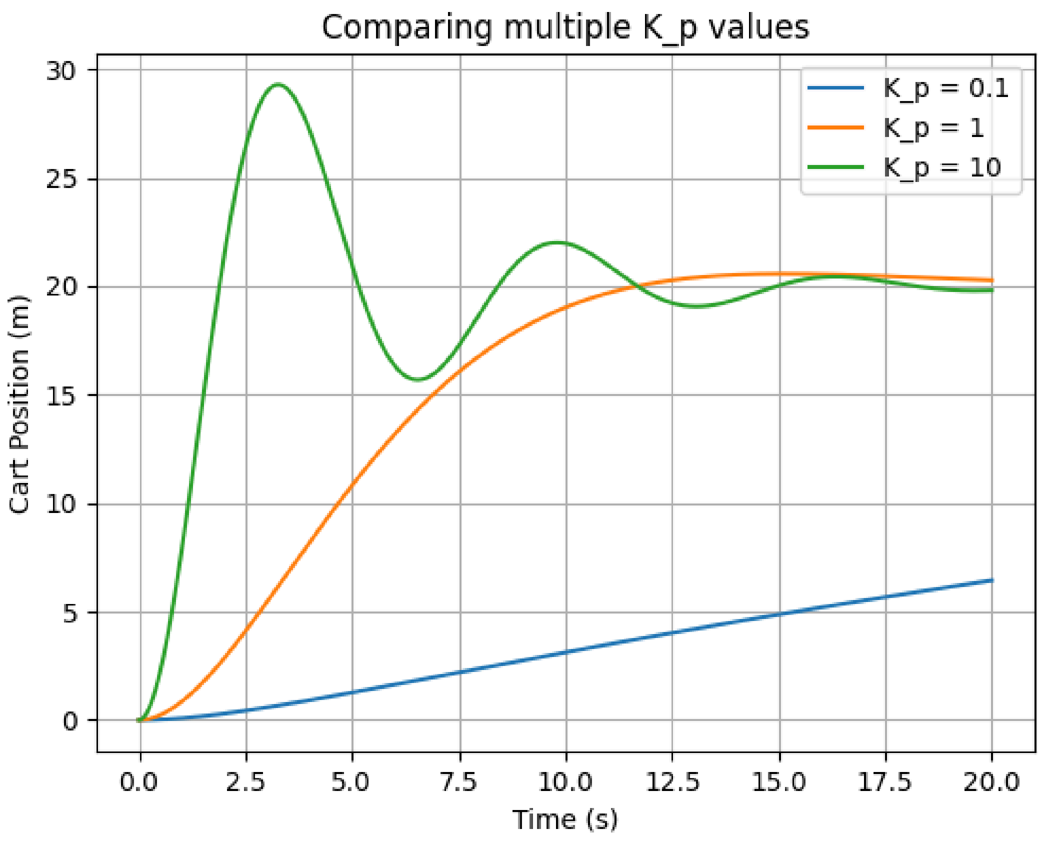

Figure 17: Comparing cart positions with different K_p values, part_3/p_compare.png

Now we shall look at the other possible parameters.

3.2 | Parameter K_i

We will simulate using $K_i = 1$, and 0 with the other parameters.

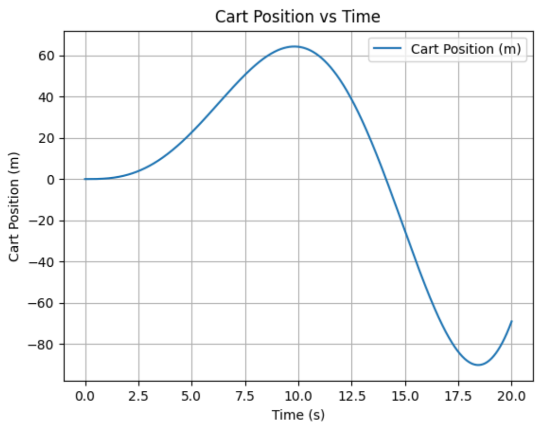

Figure 18: Cart position with K_i = 1, part_3/i1_cart_position.png

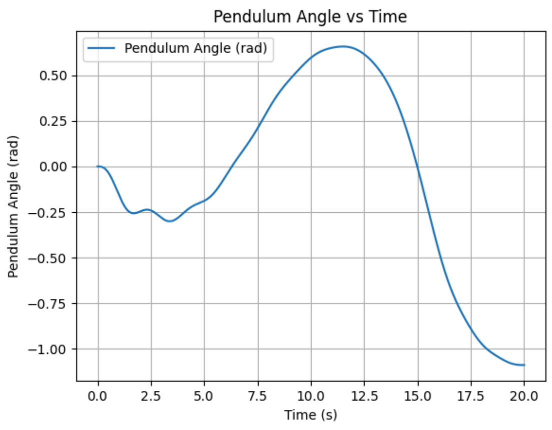

Figure 19: Pendulum angle with K_i = 1, part_3/i1_pendulum_angle.png

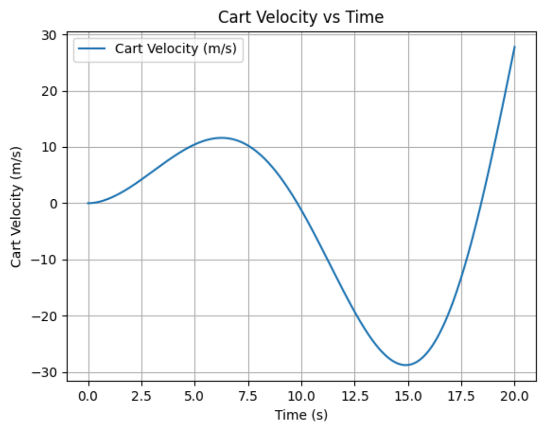

Figure 20: Cart velocity with K_i = 1, part_3/i1_cart_velocity.png

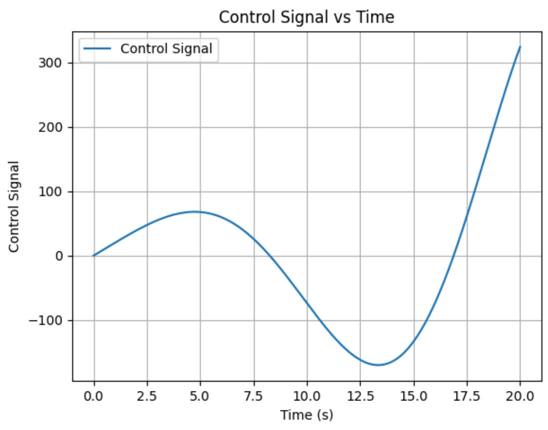

Figure 21: Control signal with K_i = 1, part_3/i1_control_signal.png

As we can see, the cart does not even try to stop at 20 meters. Instead, it stops at a place much further and then builds more speed to go in the opposite direction. This continues to happen and does not stop, the velocity will keep increasing exponentially.

u starts at 0 and slowly increases. At $t = 1$, $u \simeq 20$, where the u in K_p started. But here, u keeps increasing until x reaches 19.98215407, which is very close to 20. After that, u keeps decreasing until it reaches the local minimum, where $x = 19.99688659$, which is also very close to 20. We see a pattern here.

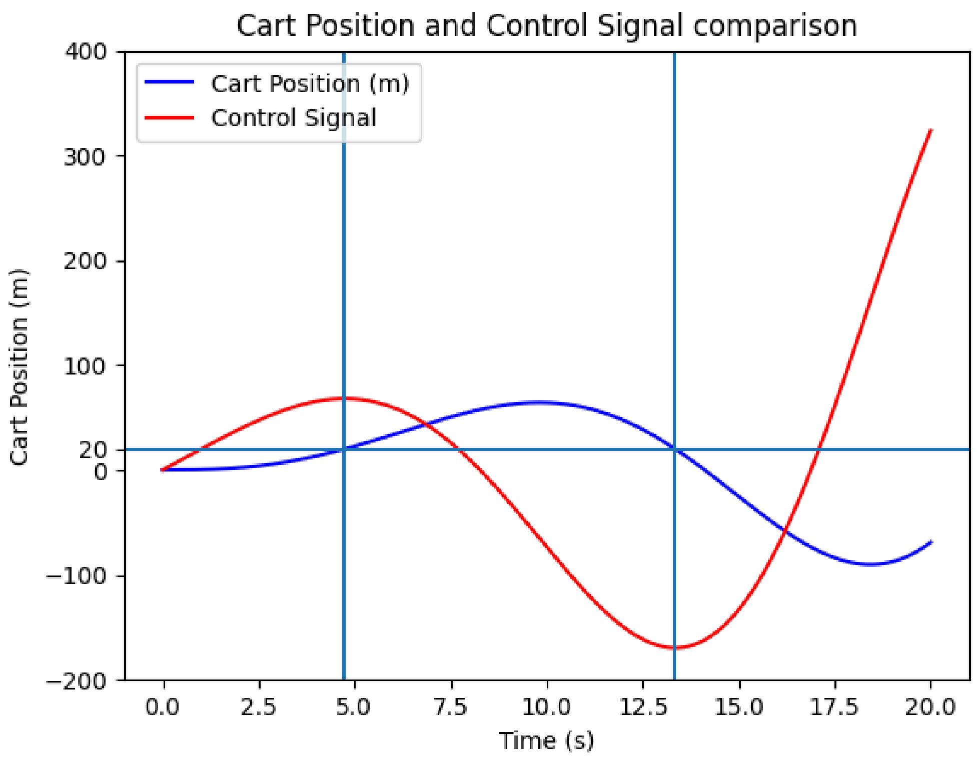

Figure 22: Comparing cart position with control signal $(K_i = 1)$, part_3/i1_position_signal.png

What this means is that the cart will only stop when it has already gone far past the destination, to the point that the starting point is closer to the destination. This results in the cart needing to have an even larger velocity to go back to the set point, which leads to a positive feedback loop, where the cart goes further and further from the set point each time it goes past it. Smaller values of K_i reduces the scale of the problem, but it is still there. Bigger values of K_i make it a lot worse. We can conclude that K_i is not a suitable parameter to work with.

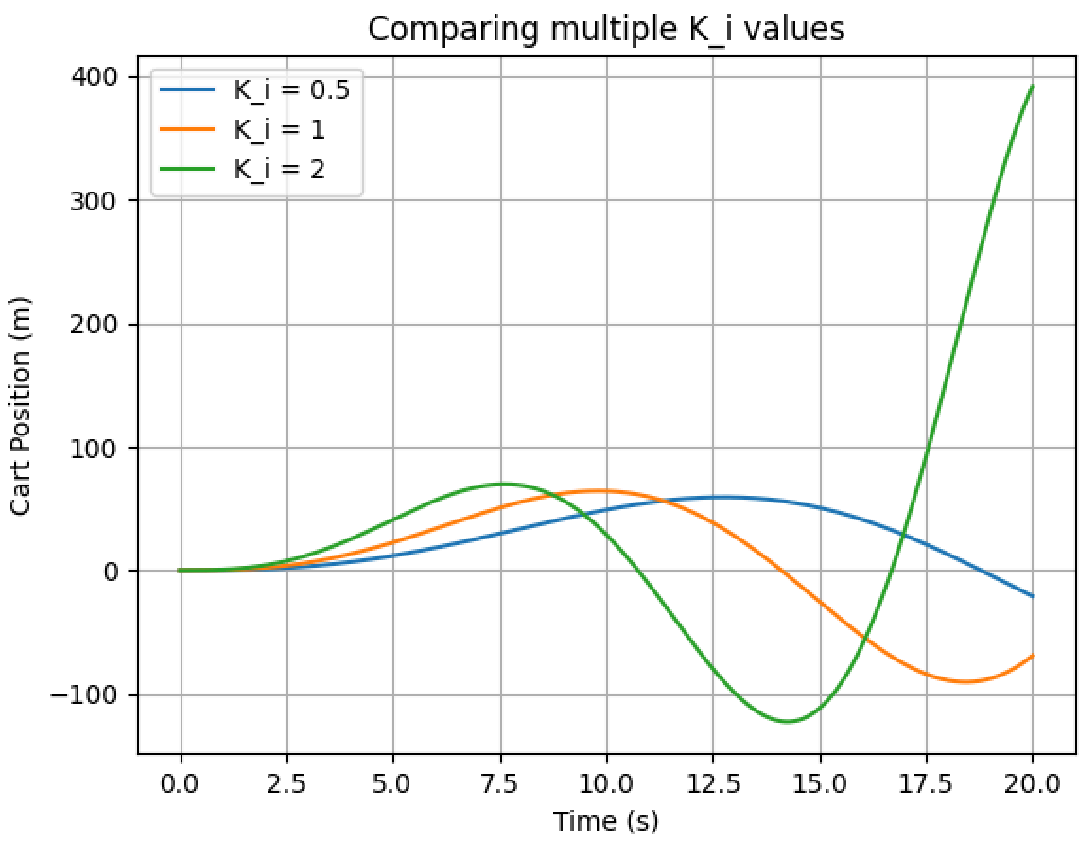

Figure 23: Cart positions with different values of K_i, part_3/i_compare.png

3.3 | Parameter K_d

We shall analyze this parameter just like the others.

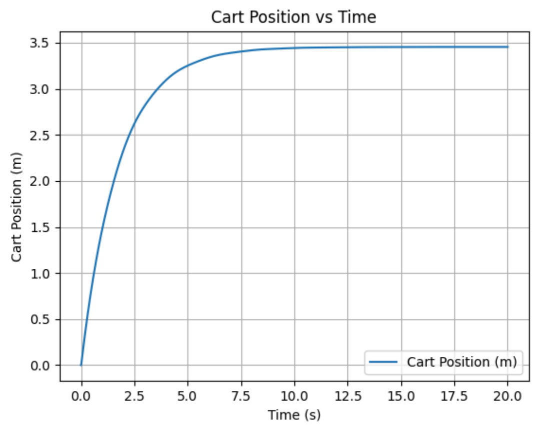

Figure 24: Cart position with K_d = 1, part_3/d1_cart_position.png

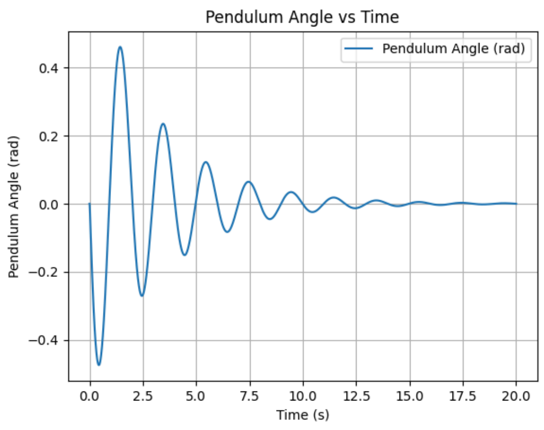

Figure 25: Pendulum angle with K_d = 1, part_3/d1_pendulum_angle.png

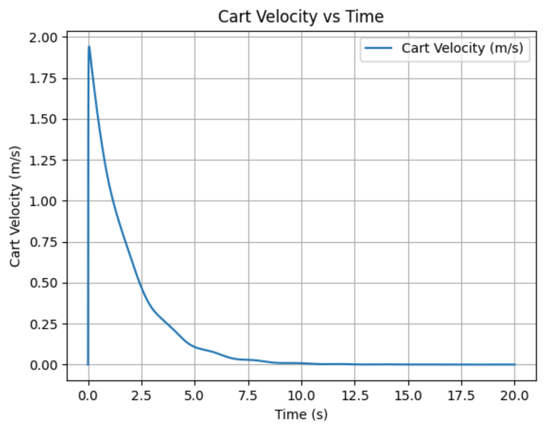

Figure 26: Cart velocity with K_d = 1, part_3/d1_cart_velocity.png

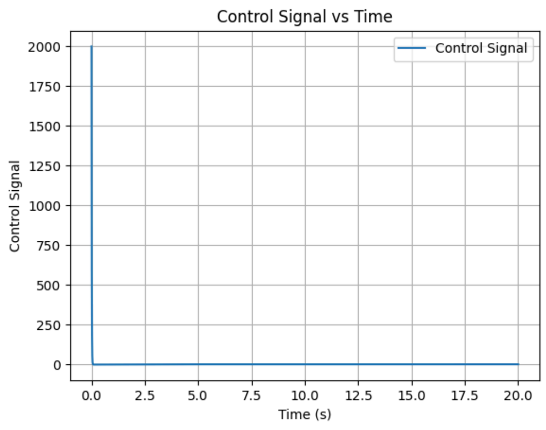

Figure 27: Control signal with K_d = 1, part_3/d1_control_signal.png

With $K_d = 10$.

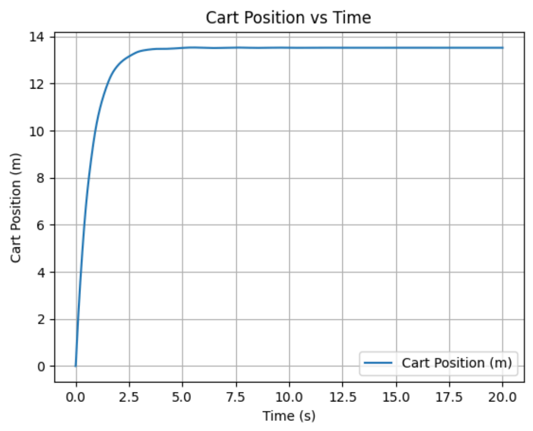

Figure 28: Cart position with K_d = 10, part_3/d10_cart_position.png

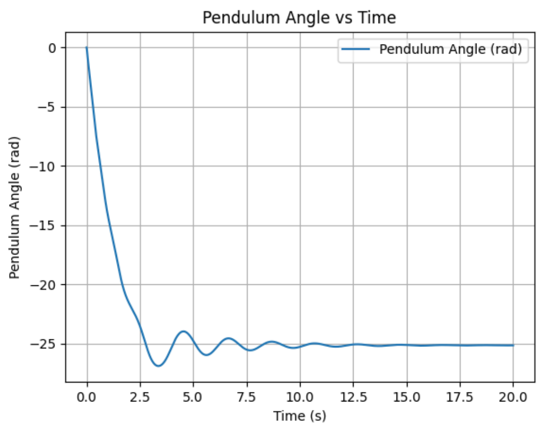

Figure 29: Pendulum angle with K_d = 10, part_3/d10_pendulum_angle.png

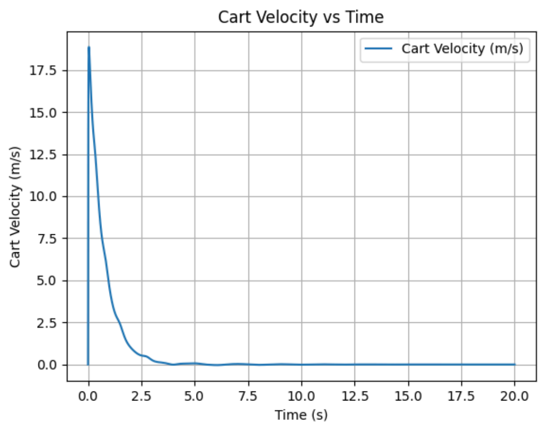

Figure 30: Cart velocity with K_d = 10, part_3/d10_cart_velocity.png

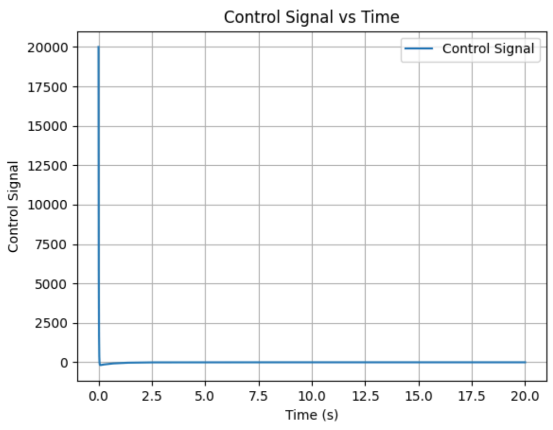

Figure 31: Control signal with K_d = 10, part_3/d10_control_signal.png

As we can see, the cart gains a burst of speed, and then slowly loses it. u Reaches an absurdly high value. The cart converges to a point, but never reaches the set point. Let's see if a higher K_d value reaches the set point.

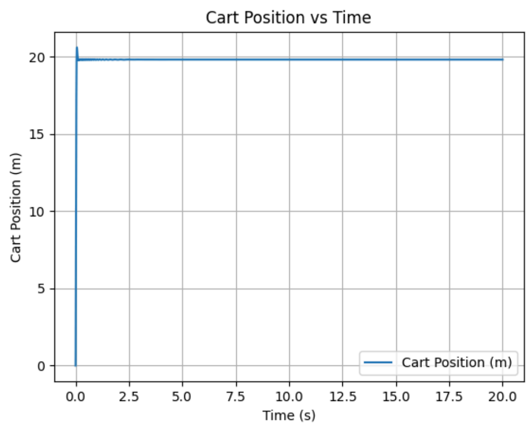

Figure 32: Cart position with K_d = 500, part_3/d500_cart_position.png

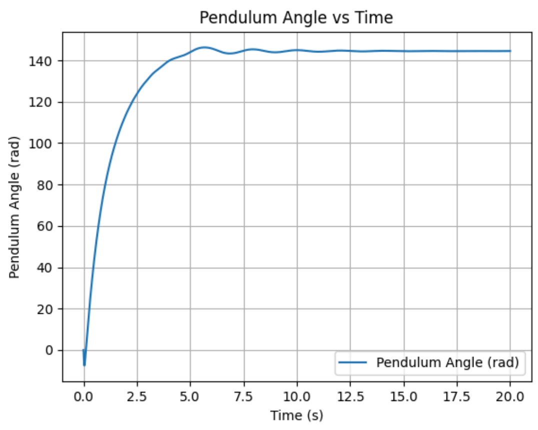

Figure 33: Pendulum angle with K_d = 500, part_3/d500_pendulum_angle.png

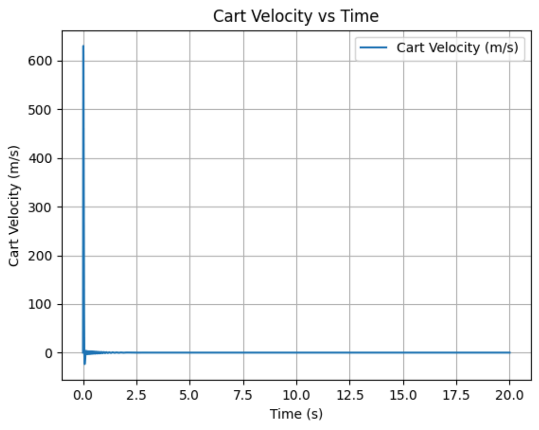

Figure 34: Cart velocity with K_d = 500, part_3/d500_cart_velocity.png

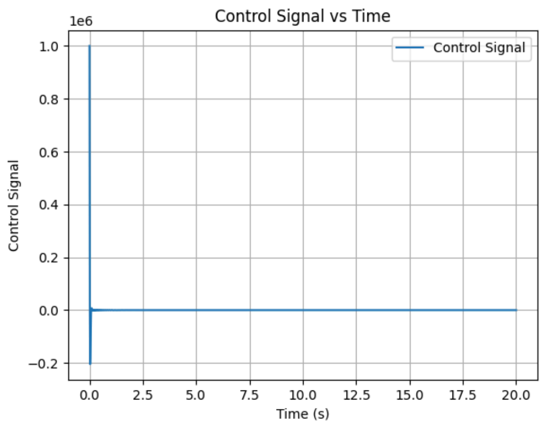

Figure 35: Control signal with K_d = 500, part_3/d500_control_signal.png

With $K_d = 500$, the cart overshoots its destination, but quickly corrects itself. So it will try to reach the set point, however with a small K_d, this never happens, and it converges at an earlier point.

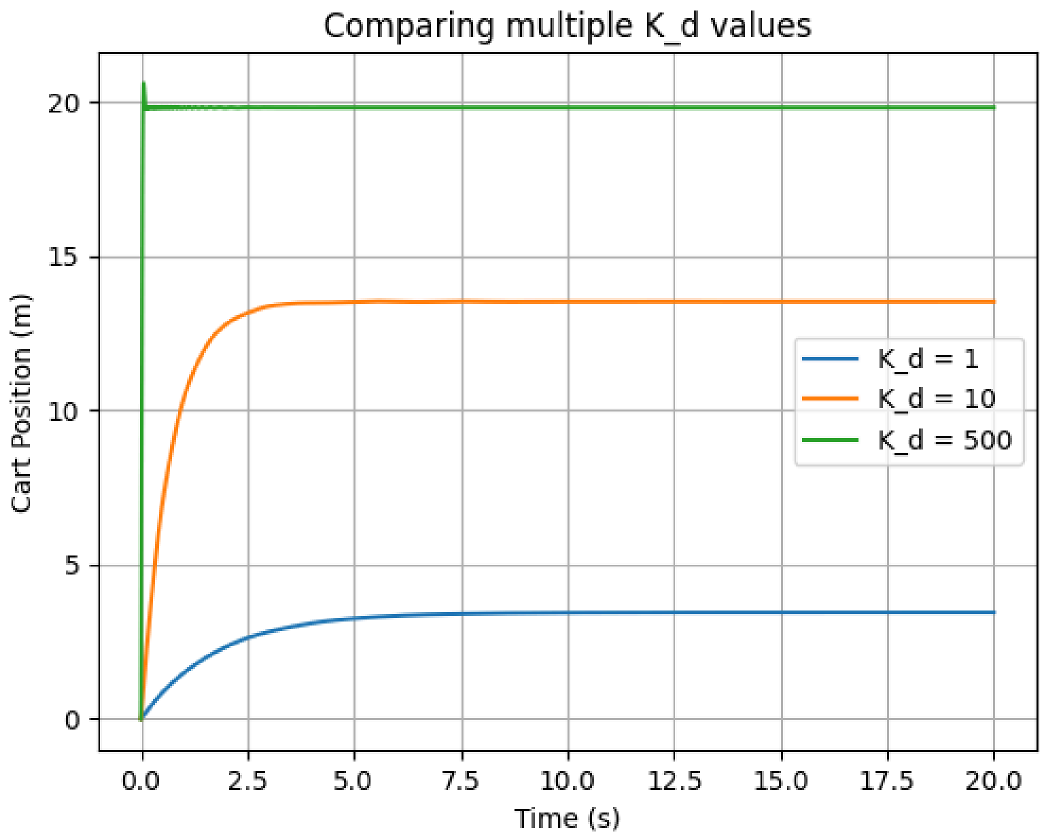

Figure 36: Cart positions with different values of K_d, part_3/d_compare.png

Higher K_d means a higher velocity and a further distance, but also makes the pendulum swing very hard. In conclusion, when using K_d alone, either it does not reach the set point, or it reaches it too fast, making it not suitable to be used alone.

3.4 | Combining parameters

After analyzing all the parameters, the best result should be using both K_p and K_d. The overshooting problem in K_p is solved with K_d, and K_d not reaching the set point is solved with K_p. K_i is useless here. The following graph shows a simulation with all parameters set to 1.

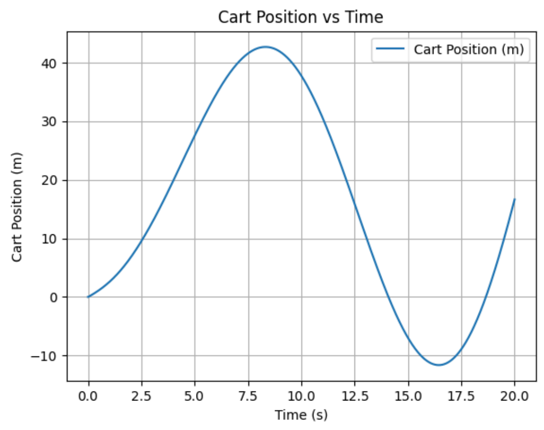

Figure 37: Cart position, all parameters 1, part_3/pid1_1_1_cart_position.png

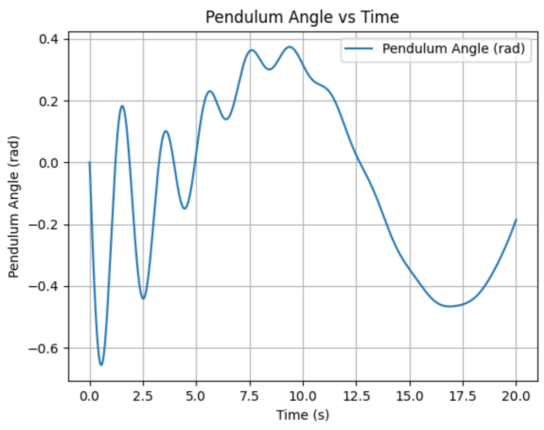

Figure 38: Pendulum angle, all parameters 1, part_3/pid1_1_1_pendulum_angle.png

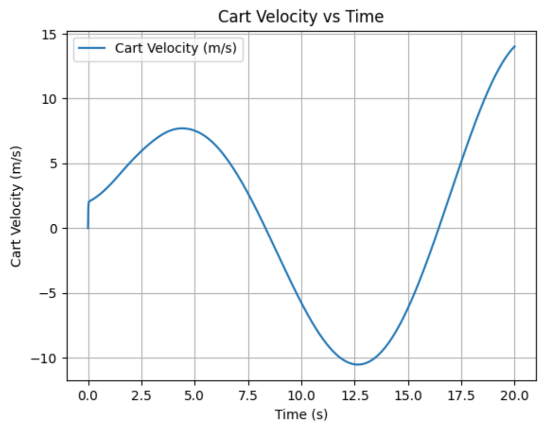

Figure 39: Cart velocity, all parameters 1, part_3/pid1_1_1_cart_velocity.png

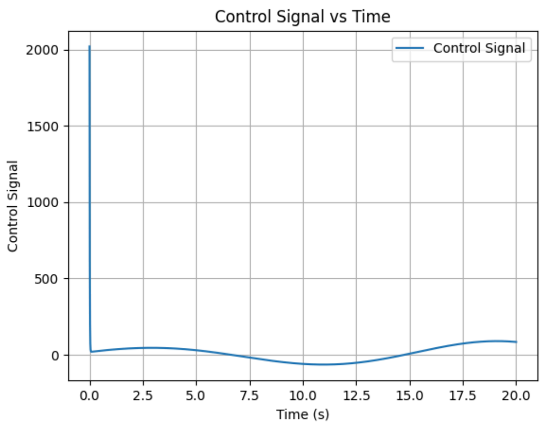

Figure 40: Control signal, all parameters 1, part_3/pid1_1_1_control_signal.png

The problem that was present with K_i, is still here. Meaning, only using K_p and K_d should optimal. The next simulation is with K_i set to 0.

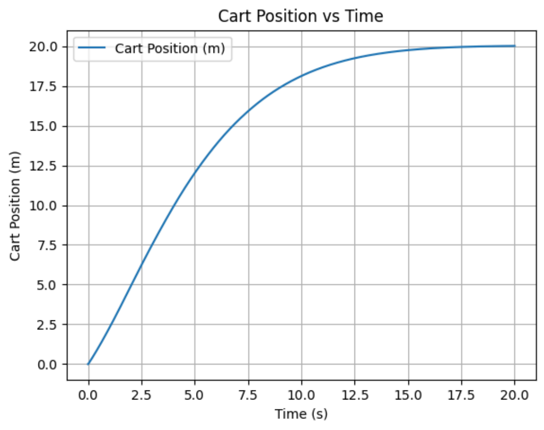

Figure 41: Cart position with K_p and K_d = 1, part_3/pid1_0_1_cart_position.png

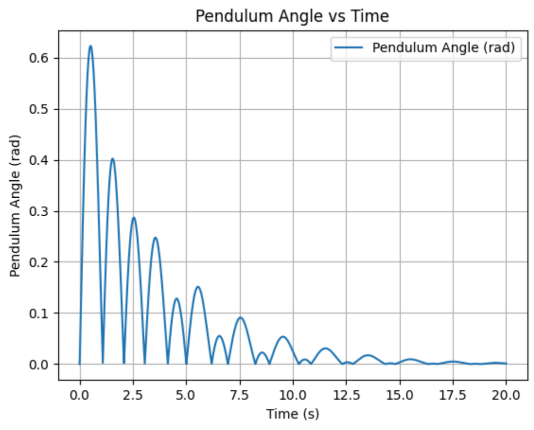

Figure 42: Pendulum angle with K_p and K_d = 1, part_3/pid1_0_1_pendulum_angle.png

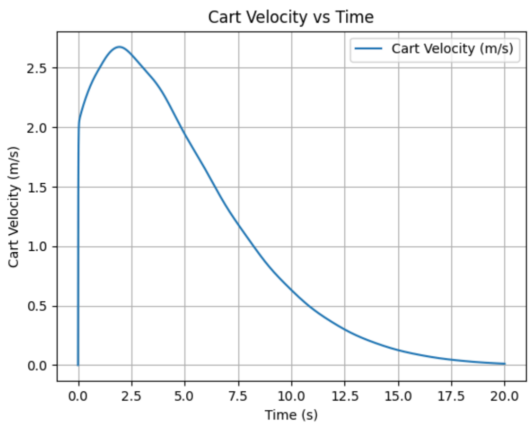

Figure 43: Cart velocity with K_p and K_d = 1, part_3/pid1_0_1_cart_velocity.png

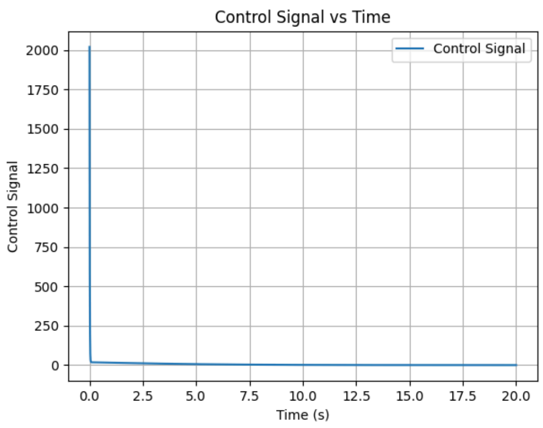

Figure 44: Control signal with K_p and K_d = 1, part_3/pid1_0_1_control_signal.png

Now it does not overshoot anymore like it does with K_p only, and also reaches the set point, what doesn't happen when only using K_d.

Now we only need to fine tune de parameters for the cart to reach the set point at a reasonable pace and without making the pendulum swing too hard. This happens in the next section.

4 | Controller Model Tuning

The last 4 digits of our student numbers are: “3082” + “2054”. We end up with a = 12 and b = 17 as the modifiers of the cost function. The cost function will be the following: $\text{cost} = 17\theta_{\text{max}} + 12t_{\text{task}}$.

In this section specifically, the set point will be 10 meters. As we only need K_p and K_d, K_i will be 0. Looping through all possible values of K_p and K_d will result in the values 16 and 10 respectively with cost 240.688. The resulting graphs look as follows:

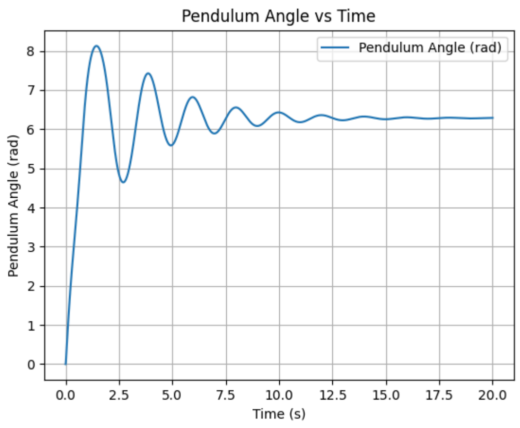

Figure 45: Pendulum angle over time, part_4/Pendulum_Angle_vs_Time.png

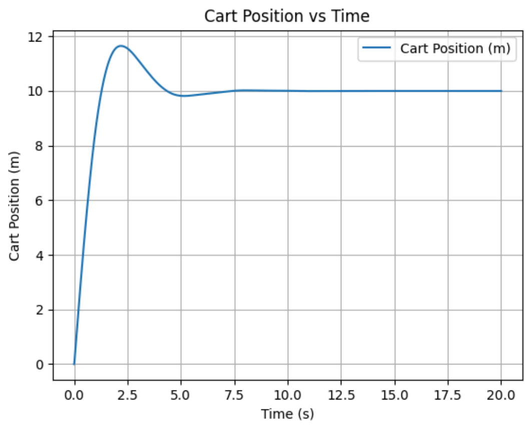

Figure 46: Cart position over time, assets/part_4/Cart_Position_vs_Time.png

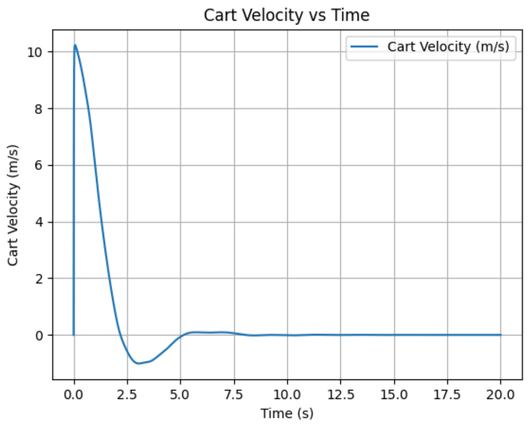

Figure 47: Cart velocity over time, assets/part_4/Cart_Position_vs_Time.png

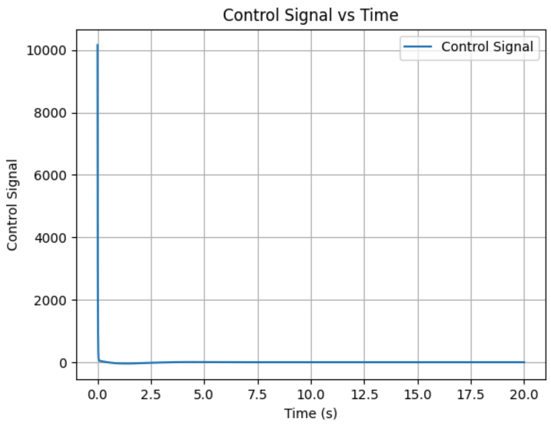

Figure 48: Cart velocity over time, assets/part_4/Cart_Position_vs_Time.png

We can see that the cart overshoots the set point and the pendulum makes a full rotation around the cart. What make sense, is that it tries to reach the set point as fast as possible, so the initial velocity is high. What does not make sense, however, is that the pendulum overshoots quite drastically even when the angular displacement has a higher weight than the time it takes in the cost function.

The pendulum overshooting can actually be explained, however. The lowest value of K_d is 10, which is already too high. Tuning the parameters again, but with K_d starting with 0, and incrementing by 1. The results are: $K_p = 4, K_d = 0$ with cost 49.51641673680321. Another reason is that the cost function is very flawed and not the best way to tune these parameters.